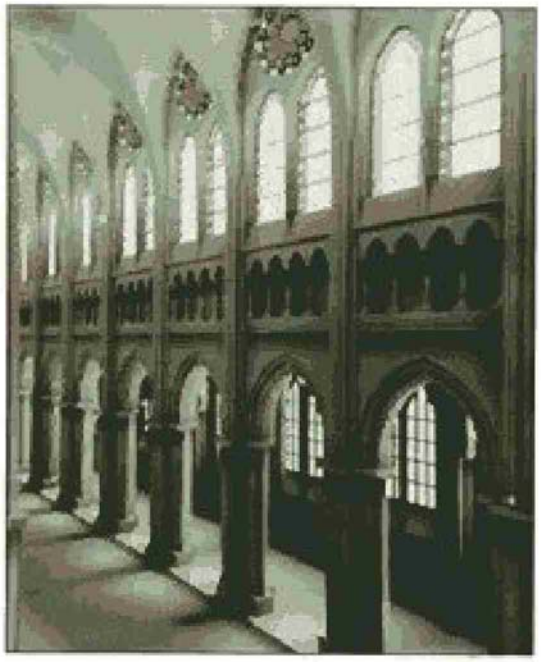
EXHIBIT O

COMPUTER GRAPHICS

C VERSION



DONALD HEARN ■ M. PAULINE BAKER

SECOND EDITION

used in a high-level programming language, such as C or FORTRAN. An example of a general graphics programming package is the GL (Graphics Library) system on Silicon Graphics equipment. Basic functions in a general package include those for generating picture components (straight lines, polygons, circles, and other figures), setting color and intensity values, selecting views, and applying transformations. By contrast, application graphics packages are designed for nonprogrammers, so that users can generate displays without worrying about how graphics operations work. The interface to the graphics routines in such packages allows users to communicate with the programs in their own terms. Examples of such applications packages are the artist's painting programs and various business, medical, and CAD systems.

Coordinate Representations

With few exceptions, general graphics packages are designed to be used with Cartesian coordinate specifications. If coordinate values for a picture are specified in some other reference frame (spherical, hyberbolic, etc.), they must be converted to Cartesian coordinates before they can be input to the graphics package. Special-purpose packages may allow use of other coordinate frames that are appropriate to the application. In general, several different Cartesian reference frames are used to construct and display a scene. We can construct the shape of individual objects, such as trees or furniture, in a scene within separate coordinate reference frames called **modeling coordinates**, or sometimes **local coordinates** or **master coordinates**. Once individual object shapes have been specified, we can place the objects into appropriate positions within the scene using a reference frame called **world coordinates**. Finally, the world-coordinate description of the scene is transferred to one or more output-device reference frames for display. These display coordinate systems are referred to as **device coordinates**, or **screen coordinates** in the case of a video monitor. Modeling and world-coordinate definitions allow us to set any convenient floating-point or integer dimensions without being hampered by the constraints of a particular output device. For some scenes, we might want to specify object dimensions in fractions of a foot, while for other applications we might want to use millimeters, kilometers, or light-years.

Generally, a graphics system first converts world-coordinate positions to **normalized device coordinates**, in the range from 0 to 1, before final conversion to specific device coordinates. This makes the system independent of the various devices that might be used at a particular workstation. Figure 2-65 illustrates the sequence of coordinate transformations from modeling coordinates to device coordinates for a two-dimensional application. An initial modeling-coordinate position (x_{mc}, y_{mc}) in this illustration is transferred to a device coordinate position (x_{dc}, y_{dc}) with the sequence:

$$(x_{mc}, y_{mc}) \rightarrow (x_{wc}, y_{wc}) \rightarrow (x_{nc}, y_{nc}) \rightarrow (x_{dc}, y_{dc})$$

The modeling and world-coordinate positions in this transformation can be any floating-point values; normalized coordinates satisfy the inequalities: $0 \leq x_{nc} \leq 1$, $0 \leq y_{nc} \leq 1$; and the device coordinates x_{dc} and y_{dc} are integers within the range $(0, 0)$ to (x_{max}, y_{max}) for a particular output device. To accommodate differences in scales and aspect ratios, normalized coordinates are mapped into a square area of the output device so that proper proportions are maintained.